MULTICORE DSP DEVICE HAVING COUPLED SUBSYSTEM MEMORY BUSES FOR GLOBAL DMA ACCESS

By:

Jay B. Reimer
Glenn C. Hopkins
Tai H. Nguyen
Yi Luo
Kevin A. McGonagle
Jason A. Jones
Duy Nguyen
Patrick J. Smith

CROSS-REFERENCE TO RELATED APPLICATIONS

Not applicable.

STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

Not applicable.

BACKGROUND OF THE INVENTION

The present invention generally relates to digital signal processors. More particularly, the invention relates to dedicated subsystem memory buses in digital signal processors. Still more particularly, the invention relates to a coupling of dedicated subsystem memory buses that allows for global memory access from any given subsystem memory bus.

Microprocessors generally include a variety of logic circuits fabricated on a single semiconductor chip. Such logic circuits typically include a processor core, memory, and numerous other support components. Some microprocessors, such as digital signal processors (DSPs) provided by Texas Instruments, may include multiple processor subsystems each having its own processor core. Each processor subsystem includes memory and other support components for the associated processor core.

DSPs are generally sought for computationally intensive tasks because they have hardware specially designed for high performance computing. The processor subsystems which may be found on multi-core DSPs typically have dedicated buses. For example, a processor subsystem may have a dedicated instruction bus that the processor core uses to retrieve program instructions from memory, a dedicated data bus that the processor core uses to retrieve data from memory, and a dedicated direct memory access (DMA) memory bus distinct from the instruction and data buses. The DMA memory bus may be a used to move data in and out of the memory without any intervention from the processor core.

The DMA memory bus in each processor subsystem typically operates under the control of an associated subsystem DMA controller. Because multiple subsystem DMA controllers exist in the DSP device, DMA data transfers between subsystems require cooperation between different DMA controllers. An efficient method for performing such transfers would be desirable.

BRIEF SUMMARY OF THE INVENTION

Accordingly, the present invention contemplates coupling of the subsystem DMA memory buses in a manner that provides each DMA controller with global DMA access to all volatile memory resources in the DSP device. In a preferred embodiment, each of the DMA controllers is coupled to each of the memory buses and is configured to control each of the memory buses. A memory bus multiplexer may be coupled between the subsystem memory bus and each of the DMA controllers, and an arbiter may be used to set the memory bus multiplexer so as to allow any one of the DMA controllers to control the memory bus. The memory bus may also be controlled by the host port interface via the memory bus multiplexer. A round-robin arbitration technique is used to provide each of the DMA controllers and the host port interface fair access to the memory bus. This approach may advantageously provide increased flexibility in the use of DMA controllers to transfer data from place to place, with only a minimal increase in complexity.

BRIEF DESCRIPTION OF THE DRAWINGS

For a detailed description of the preferred embodiments of the invention, reference will now be made to the accompanying drawings in which:

Figure 1 shows a DSP device having subsystem DMA buses coupled together;

Figure 2 shows an alternative configuration for coupling the DMA buses together; and

Figure 3 shows a high-level state machine diagram of a memory bus arbiter.

NOTATION AND NOMENCLATURE

Certain terms are used throughout the following description and claims to refer to particular system components. As one skilled in the art will appreciate, semiconductor companies may refer to a component by different names. This document does not intend to distinguish between components that differ in name but not function. In the following discussion and in the claims, the terms "including" and "comprising" are used in an open-ended fashion, and thus should be interpreted to mean "including, but not limited to...". Also, the term "couple" or "couples" is intended to mean either an indirect or direct electrical connection. Thus, if a first device couples to a second device, that connection may be through a direct electrical connection, or through an indirect electrical connection via other devices and connections.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The preferred embodiment of the present invention is discussed below in the context of a multi-core, fixed-point, digital signal processor (DSP) chip. This embodiment, however, is not intended to limit the scope of this disclosure to this context, rather, the preferred embodiment may have applicability to any multiple core DSP device that would benefit from global DMA access.

Turning now to the figures, Figure 1 shows a DSP chip 100 that includes multiple DSP subsystems 110, 120, a shared program memory (PRAM) 132, a memory bus interface 134, an external I/O port (XPORT) arbiter 136, an XPORT multiplexer 138, and a host port interface (HPI) multiplexer 139. Each DSP subsystem 110, 120 (generally

separated by the dashed line in Figure 1) preferably includes a DSP core 11, 21, a read-only memory (ROM) 12, 22, a dual-access, random access memory (DARAM) 13, 23, a single-access, random access memory (SARAM) 14, 24, one or more peripheral devices 15, 25, an M-bus multiplexer 16, 26, an M-bus arbiter 17, 27, a DMA controller 18, 28, a host port interface (HPI) 19, 29, and other miscellaneous support circuitry. The subsystems 110, 120 each further include an instruction bus P1, P2, a data bus D1, D2, a memory bus M1, M2, a processor core external I/O bus XC1, XC2, and a DMA controller external I/O bus XD1, XD2.

The shared program memory (PRAM) 132 preferably is reserved for program instructions, and includes 16 blocks of dual-access RAM. Each block comprises 16 kilobytes of storage, although the block size and number of blocks can be varied as desired. Each DSP subsystem 110, 120 can fetch an instruction from any location in the PRAM 132 during each clock cycle. The processor cores 11, 21 concurrently fetch and execute distinct instructions from a single program stored in the PRAM 132. Although the DSP cores may execute the same software program, they do not necessarily execute the same instructions concurrently or necessarily follow the same branches in program flow.

According to the preferred embodiment, the DSP cores 11, 21 are not permitted to write to the PRAM 132. Instead, a host processor (not shown) provides the software to the PRAM 132 via the XPORT, HPI 19, 29 and memory buses M1, M2 as described further below.

The memory bus interface 134 is coupled to PRAM 132 and to the memory buses M1, M2. The memory bus interface 134 provides a set of first-in, first-out (FIFO) buffers that the memory buses M1, M2 can write to and read from. Each FIFO buffer is one way, that is, written to by one memory bus and read by the other. This provides one method of inter-subsystem communication. The memory bus interface 134 also couples both memory buses M1, M2 to PRAM 132. The memory bus interface includes an arbiter which grants one of the memory buses access to PRAM when such accesses are sought. The initial programming of the PRAM and updates of the PRAM are typically performed via the memory buses.

The XPORT arbiter 136 and XPORT multiplexer 138 are coupled to the processor cores 11, 21 and the DMA controllers 18, 28 in each of the subsystems via respective

external I/O buses XC1, XC2, XD1, XD2. The processor cores and DMA controllers arbitrate for external access as explained further below, and the arbiter 136 sets the multiplexer 138 in accordance with the arbitration results. The DSP 100 is provided in a semiconductor package that has multiple pins ("leads") to provide external connections for the chip. The package leads used by the XPORT for external access are preferably shared with the host port interface units 19, 29. Accordingly, the output from XPORT multiplexer 138 is coupled to the HPI multiplexer 139, as are the HPI units 19, 29. When the host processor asserts the MODE signal (which is the control signal for the HPI multiplexer 139) the XPORT pins are coupled to the HPI units 19, 29, and the host processor accesses the DSP device 100 as a memory-mapped device. When the host processor de-asserts the MODE signal, the XPORT leads are coupled to the XPORT multiplexer 138, and any external accesses are initiated by the cores 11, 21 or the DMA controllers 18, 28, as explained further below.

The processor cores 11, 21 preferably execute software instructions retrieved via corresponding instruction buses P1, P2 to operate on data retrieved via corresponding data buses D1, D2. Results are returned from the processor cores on the data buses. The processor cores typically include an optimized arithmetic logic unit (ALU) and a control unit. The control unit retrieves data and instructions and decodes the instructions, and the ALU operates on the data as specified by the instructions.

The ROMs 12, 22 are non-volatile memories coupled to the corresponding instruction buses P1, P2. The ROMs preferably store boot-up software for initializing the subsystems. The DARAMs 13, 23 preferably include four memory blocks, each of which support two memory accesses per clock cycle. The DARAMs 13, 23 are intended primarily for data storage, but may be used to store program instructions as well. Accordingly, they are coupled to both the corresponding instruction buses P1, P2 and to the corresponding data buses D1, D2. A register (not shown) in the DSP core 11, 21 determines whether the DARAM 13, 23 is mapped into program memory space or data memory space. The SARAMs 14, 24 preferably also include four memory blocks, each of which support one memory access per clock cycle. Each SARAM preferably is reserved for data storage, and accordingly is coupled to the corresponding data bus D1, D2.

Referring still to Figure 1, instruction buses P1, P2 couple together the corresponding processor core 11, 21, the local DARAM 13, 23, the local ROM 12, 22, and the shared PRAM 132. Data buses D1, D2 couple together the corresponding processor core 11, 21, the local DARAM 13, 23, and the local SARAM 14, 24. Memory buses M1, M2 couple the memory bus multiplexer 16, 26 with each of the volatile memory devices 13, 14, 23, 24, 132 in the corresponding subsystem. The memory buses also couple to peripheral devices 15, 25.

Peripheral devices 15, 25 preferably each include one or more multi-channel, serial interfaces. The multi-channel serial interfaces provide high-speed, full-duplex, double-buffered serial communications. The configuration of these ports is preferably programmable by the associated processor core to allow direct interfacing with existing serial protocols. Each serial interface 15, 25 preferably supports multi-channel transmit and receive of up to 128 channels. The multi-channel serial ports perform time division multiplexing and de-multiplexing when multiple channels are enabled. Each data frame that is sent or received represents a time-division multiplexed (TDM) data stream, so that the content of one channel is interleaved with the contents of the other channels.

Memory bus multiplexers 16, 26 and memory bus arbiters 17, 27 are each coupled to all DMA controllers 18, 28 and HPI units 19, 29. The local DMA controller 18, the local HPI unit 19, the remote DMA controller 28, and the remote HPI unit 29 can each control memory bus M1 via memory bus multiplexer 16 to access peripherals 15, SARAM 14, DARAM 13, and PRAM 132. Similarly, each of them can control memory bus M2 via memory bus multiplexer 26 to access peripherals 25, SARAM 24, DARAM 23, and PRAM 132. Accordingly, each of the DMA controllers has global access, as does each of the HPI units. Arbitration among the local DMA controller, the local HPI unit, and the remote subsystem for access to memory bus M1 is performed by arbiter 17, which then sets the memory bus multiplexer 16 in accordance with the arbitration winner. Multiplexer 26 and arbiter 27 operate similarly for accesses via memory bus M2.

Each DMA controller 18, 28 moves data and instructions to and from local peripherals and data storage devices, and to shared PRAM 132, via the corresponding memory bus M1, M2. Each DMA controller 18, 28 can also move data to and from remote peripherals and data storage devices via the remote memory bus. Finally, each DMA

controller can move data to and from external sources via an external I/O bus XD1, XD2 and the XPORT. Although the transfers may be initiated in different ways, including initiation by the processor core, the transfers are thereafter performed "in the background", i.e., without active monitoring and control by the processor core. Each DMA controller preferably provides multiple "channels" for the independent, concurrent management of multiple block transfers. DMA transfers are accomplished by first reading the data into memory internal to the DMA controller, and then writing the data from the DMA controller memory to the desired destination. When processor core memory accesses to internal memory conflict with DMA controller accesses, the DMA controller accesses are preferably given higher priority.

The HPI units 19, 29 allow an external host processor to access all internal memory via the memory buses M1, M2. To keep the overall system design simple, the host processor interfaces 19, 29 are designed to mimic a memory interface. That is, the host processor can "view" the contents of any memory location internal to the DSP device 100 and many of the processor core registers by sending an address to the HPI units 19, 29 indicating the desired location. One of the HPI units 19, 29 then retrieves the desired information and provides the information as data in the same way that a memory device would. The HPI units 19, 29 can similarly store data in the desired location. The software to be executed by the processor cores may be provided by the host processor in this manner. That is, the host processor may write the software to shared PRAM 132 via the HPI 19, 29. The HPI units 19, 29 preferably act as a slave device to the host processor, but may generate a signal to the host processor to stall the host processor during an access if the memory buses M1, M2 are busy with other tasks.

Figure 2 shows an alternative embodiment for coupling the DMA controllers 18, 28 and HPI units 19, 29 to the memory buses M1, M2. Remote-access multiplexers 62 and remote access arbiters 64 have been added. If the local DMA controller or local HPI unit (e.g. 18, 19) seeks access to a remote memory bus (e.g., M2), a remote access arbiter 64 detects the access request and sets a remote access multiplexer 62 accordingly. The remote access arbiters 64 resolve conflicts on a rotating priority basis. That is, if the remote DMA controller wins an access conflict with the remote HPI unit in a given clock cycle, the

remote HPI will be given priority the next time a conflict occurs with the remote DMA controller.

The output of the remote access multiplexer 62 is received by the remote memory bus arbiter and multiplexer (e.g., 26, 27). The memory bus arbiter (e.g. 27) arbitrates between its local DMA controller (e.g. 28), its local HPI unit (e.g. 29), and the remote access via multiplexer 62, and sets the memory bus multiplexer in accordance with the arbitration winner.

Each of the multiplexers 16, 26, 62 preferably grants only one access at a time. The accesses which are not immediately granted will be granted in due course. Accordingly, the DMA controllers and HPI units simply maintain their access attempts until access is granted.

Figure 3 shows an illustrative high-level state diagram that may be implemented by memory bus arbiters 17, 27. In the absence of any attempted memory bus accesses, the memory bus arbiter continuously and sequentially checks for local DMA access requests 42, HPI access requests 44, and remote access requests 46. The local DMA access requests come from the local DMA controller, HPI access requests are made by the local HPI unit, and remote access requests may come from a remote access multiplexer 62 or alternatively directly from a remote DMA controller or HPI unit. If no local DMA access request is detected, the memory bus arbiter 17, 27 checks for HPI access requests 44. If no HPI access request is detected, the memory bus arbiter checks for remote DMA access requests 46. If no remote DMA access request is detected, the memory bus arbiter again checks for local DMA access requests 42. The memory bus arbiter 17, 27 checks the various access request sources sufficiently rapidly to initiate a memory bus access the clock cycle after it is received, assuming that the requested access wins this round-robin arbitration scheme.

If the memory bus arbiter 17, 27 detects a local DMA access request, the memory bus arbiter sets the memory bus multiplexer 16, 26 and allows the DMA controller 18, 28 to perform a memory bus transaction 48. The DMA controller normally transfers data in two steps: a read from the source to internal memory in the DMA controller, followed by a write from the internal memory to the desired destination. The memory bus transaction may accordingly be a read or a write. The read step and the write step of a DMA transfer may be separated by other memory bus transactions, e.g. an HPI transaction 50 or a remote

access transaction. After the DMA memory bus transaction is completed, the memory bus arbiter resumes checking, beginning with HPI access requests 44.

If the DMA controller 18, 28 detects an HPI access request 44, the DMA controller performs the HPI transaction 50. Again, the transaction may be a read access or a write access. In a read access, the HPI unit retrieves information requested by a host processor. In a write access, the HPI unit stores information from the host processor in the desired location. After the transaction is completed, the memory arbiter resumes checking, beginning with the remote DMA access requests 46.

If the memory arbiter 17, 27 detects a remote access request, the memory arbiter allows the remote DMA controller or remote HPI unit (via the remote access multiplexer) to perform a remote access transaction 52 on the memory bus. The transaction may be a read access or a write access performed in a manner similar to that described above. After completion of the transaction, the memory arbiter resumes checking, beginning with local DMA requests 42.

In the embodiments of Figures 1 and 2, the DSP chip 100 includes only two DSP subsystems 101, 102. As one skilled in the art will appreciate, there may be more than two DSP subsystems, each having a corresponding processor core.

The above discussion is meant to be illustrative of the principles and various embodiments of the present invention. Numerous variations and modifications will become apparent to those skilled in the art once the above disclosure is fully appreciated. It is intended that the following claims be interpreted to embrace all such variations and modifications.